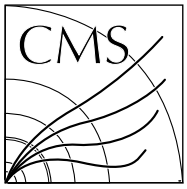
Available on CMS information server

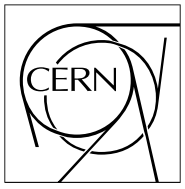
CMS NOTE 2007/008



The Compact Muon Solenoid Experiment

CMS Note

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland



26 March 2007

Adaptive Vertex Fitting

R. Frühwirth, W. Waltenberger

Institute of High Energy Physics of the Academy of Sciences, Vienna, Austria

P. Vanlaer

IIHE (ULB-VUB), Pleinlaan 2, B-1000 Brussels, Belgium

Abstract

Vertex fitting frequently has to deal with both mis-associated tracks and mis-measured track errors. A robust, adaptive method is presented that is able to cope with contaminated data. The method is formulated as an iterative re-weighted Kalman filter. Annealing is introduced to avoid local minima in the optimization. For the initialization of the adaptive filter a robust algorithm is presented that turns out to perform well in a wide range of applications. The tuning of the annealing schedule and of the cut-off parameter is described, using simulated data from the CMS experiment. Finally, the adaptive property of the method is illustrated in two examples.

1 Introduction

The method of Least Squares is seen to be our best course when we have thrown overboard a certain portion of our data – a sort of sacrifice which has often to be made by those who sail the stormy seas of Probability.

F. Y. Edgeworth, 1887

Vertex fitting is the task of computing the location and the error of an interaction vertex from a given set of reconstructed tracks. A widely used method for this purpose is the Kalman filter [1, 2] which was implemented in the CMS reconstruction program ORCA [3] and is now available in the new framework CMSSW [4]. The Kalman filter is a least-squares estimator which minimizes the sum of the squared standardized distances of all tracks from the vertex position $\boldsymbol{v}$:

$$\hat{\boldsymbol{v}}_{\mathrm{LS}} = \underset{\boldsymbol{v}}{\operatorname{argmin}} L(\boldsymbol{v}), \quad \text{with} \quad L(\boldsymbol{v}) = \frac{1}{2}\sum_{i=1}^{n} \chi_i^2(\boldsymbol{v}) = \frac{1}{2}\sum_{i=1}^{n} d_i^2(\boldsymbol{v})/\sigma_i^2. \tag{1}$$

Differentiation with respect to $\boldsymbol{v}$ gives the following equation for $\hat{\boldsymbol{v}}$:

$$\frac{\partial L(\boldsymbol{v})}{\partial \boldsymbol{v}} = \sum_{i=1}^{n} \chi_i(\boldsymbol{v}) \frac{\partial \chi_i}{\partial \boldsymbol{v}} = 0. \tag{2}$$

Usually the distance d_i is approximated by an affine function of $\boldsymbol{v}$, using a first-order Taylor expansion:

$$d_i(\boldsymbol{v}) \approx c_i + \boldsymbol{a}_i^T \boldsymbol{v}. \tag{3}$$

Equation (2) then becomes a linear equation for $\hat{\boldsymbol{v}}$ and can be solved explicitely, either globally or iteratively with the Kalman filter.

Least-squares estimators are known not to be robust, which means that they are sensitive to contaminated data, such as mis-associated tracks or mis-measured track errors. In one of the authors' PhD thesis [5] a few robustifications of the standard Kalman filter have been suggested, one of which has turned out to be a very powerful general-purpose technique: the adaptive vertex fitter (AVF). This paper deals almost exclusively with this most successful method. Techniques which have turned out to be less powerful are only hinted at; the more interested reader is referred to the aforementioned thesis. While this paper is intended to describe the method and motivate its default values, another CMS note [6] systematically compares the AVF against the classical methods.

2 The adaptive vertex fitter

The adaptive vertex fitter does not reject an outlying track; rather it down-weights the outlier with a weight w_i [7, 8]. The weight w_i depends on the compatibility of track i with the vertex, as measured by χ_i^2:

$$w_i\left(\chi_i^2\right) = \frac{\exp(-\chi_i^2/2T)}{\exp(-\chi_i^2/2T) + \exp(-\chi_{\mathrm{c}}^2/2T)}. \tag{4}$$

The weight w_i can be interpreted as the probability that track i belongs to a vertex at $\boldsymbol{v}$. The constant χ_{c}^2 defines the threshold where the weight is equal to $1/2$; beyond this threshold a track is considered to be more likely an outlier than an inlier. The temperature T is a parameter that controls the shape of the functional dependence in Eq. (4). A zero temperature results in a step function and is equivalent to a hard cut at χ_{c}^2. Figure 1 shows the weight as a function of χ, with a cutoff at $\chi_{\mathrm{c}} = 3$, for three different temperatures.

After including the weights the fit equation (Eq. (2)) reads

$$\sum_{i=1}^{n} w_i(\chi_i^2(\boldsymbol{v}))\chi_i(\boldsymbol{v}) \frac{\partial \chi_i}{\partial \boldsymbol{v}} = 0. \tag{5}$$

The weight of track i is now reduced by a factor $w_i(\chi_i^2)$. As the weights depend on the vertex position $\boldsymbol{v}$, an iterative procedure is required to solve Eq. (5). The weights are computed for an initial vertex position, and the vertex is estimated using these weights. These two steps are repeated until convergence. The resulting estimator can be regarded as an M-estimator [9], the result of minimizing an objective function of the form

$$M(\boldsymbol{v}) = \sum_{i=1}^{n} \rho(\chi_i(\boldsymbol{v})). \tag{6}$$

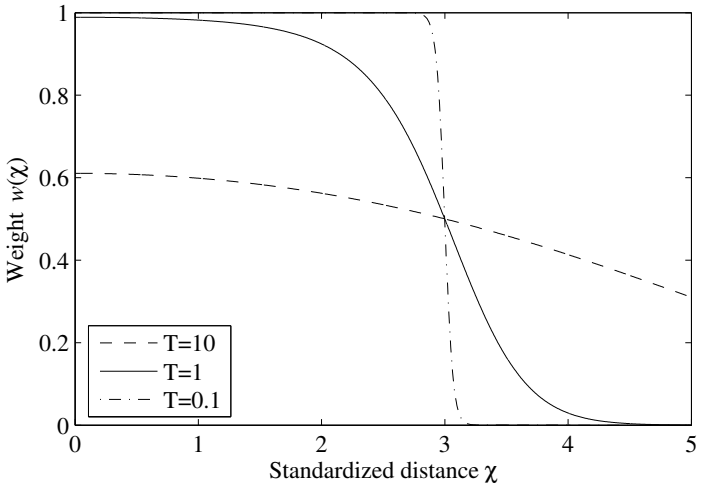


Figure 1: The weight function of Eq.(4) at three different temperatures.

In the special case $\rho(\chi_i) = \chi_i^2$, the least-squares estimator is recovered. Obviously the M-estimator is a solution of the equation

$$\frac{\partial M(\boldsymbol{v})}{\partial \boldsymbol{v}} = \sum_{i=1}^{n} \psi(\chi_i(\boldsymbol{v})) \frac{\partial \chi_i}{\partial \boldsymbol{v}} = 0, \quad \text{with} \quad \psi(t) = \rho'(t). \tag{7}$$

A comparison with Eq. (5) shows that with our choice of weights

$$\psi(\chi) = \chi \frac{\exp(-\chi^2/2T)}{\exp(-\chi^2/2T) + \exp(-\chi_c^2/2T)}. \tag{8}$$

As ψ vanishes for the limit of large χ, the M-estimator is of the redescending type [9]. Integrating ψ over χ yields the ρ-function of the adaptive estimator:

$$\rho(\chi) = \frac{1}{2}\chi^2 - T \ln\left(\exp(\chi^2/2T) + \exp(\chi_c^2/2T)\right) + T \ln\left(1 + \exp(\chi_c^2/2T)\right). \tag{9}$$

The constant of integration has been chosen such that $\min \rho = 0$. Figure 2 shows the shape of the function $\rho(\chi)$ for three different values of the temperature T, with the same cut ($\chi_c = 3$) as in Fig. 1. If the temperature is at $T = 1$, the M-estimator is very close to a least-squares estimator for tracks within the cut, whereas for tracks beyond the cut, the contribution to the objective function is nearly constant. As a consequence, the vertex position is influenced very little by the outliers.

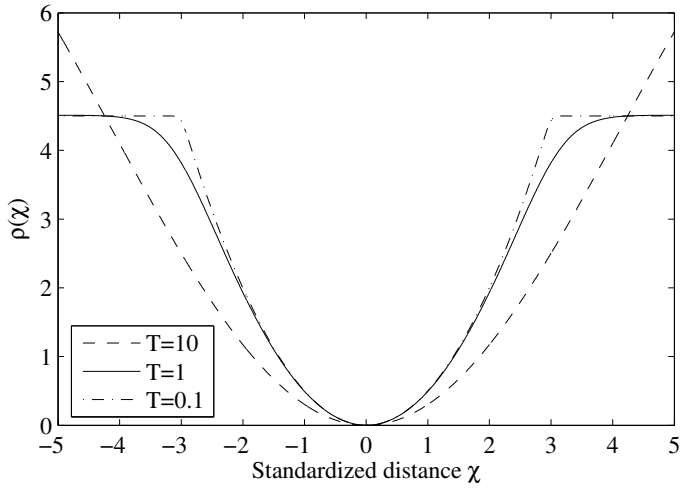


Figure 2: The function ρ of Eq. (9) at three different temperatures.

As mentioned before, the definition of the weights in Eq. (4) introduces the notion of a temperature T. This temperature can be used to employ a deterministic annealing schema that helps to avoid falling into local minima. The estimation starts at a user-defined initial temperature $T_{\mathrm{ini}} > 1$. The temperature is then lowered in each step in a well-defined sequence that converges to 1. The iteration is stopped as soon as:

- the temperature is equal to 1, and
- the vertex candidate position has not changed by more than one micron.

The implementation of the adaptive vertex fitter method is straightforward, given a Kalman filter implementation that accounts for the notion of track weights. Details of the implementation are given in the appendix. An example of an adaptive vertex fit is visualized in Fig. 3.

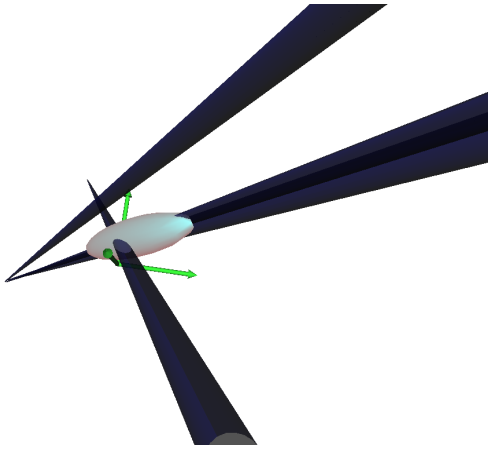


Figure 3: Result of an adaptive fit. The fitter was supplied with four tracks ($K^+K^-\mu^+\mu^-$), one of which is incompatible with the other three. Two tracks are highly collimated and appear as one in the plot. The fitter completely ignores the outlying track. The size of the ellipsoid has been multiplied by a factor of ten. The three arrows behind the vertex have a length of 100 μm in the "ellipsoid scale", and a length of $1\,\mathrm{mm}$ in the "track scale".

3 Test samples

All case studies described in this paper have been performed with ORCA version 8.2.0. All events are without any simulated pile-up. If not stated otherwise, the default parameters of the adaptive fitter are used. Track reconstruction is performed by ORCA's default track reconstruction method. Four different kinds of event topologies are considered:

- $c\bar{c}$ jets: high multiplicity events. This topology implements a benchmark for fitting primary vertices with secondary vertices as a "background". The transverse jet energy is 100 GeV, and the jets are in the tracker barrel region ($|\eta| < 1.4$). The primary vertex is fitted using all tracks within the jet cone found by the PersistentJetFinder (with default values).
- $q\bar{q}$ jets: similar to the $c\bar{c}$ case, but there are fewer secondary vertices with fewer tracks and a higher average distance to the primary vertex. Also, the primary vertices tend to contain more tracks. Again, the jet transverse energy is 100 GeV, in the barrel region only. The primary vertex is, again, fitted using all tracks within the jet cone.
- $\tau \to \pi^\pm\pi^\pm\pi^\mp$: a 3-prong vertex that will be a good benchmark for fitting highly collimated low-multiplicity secondary vertices. Contamination comes from mis-measured tracks. Tracks matching the simulated pions from the τ-decay have been selected. The events have been obtained by producing a light MSSM Higgs $h^0 \to \tau^+\tau^- \to 6\,\pi$, and selecting three-prong τ decays.
- $B_s \to J/\psi\ \varphi \to KK\mu\mu$: a 4-prong vertex (if reconstructed correctly) that will serve as another secondary vertex fitting test case. The simulated events were preselected such that both muons have a $p_T > 2$ GeV/c. Again, the data contains mis-measured tracks. It does not contain mis-associated tracks.

The rest of this section will give a few details of the event characteristics in the four test samples. For the remainder of this section, the following applies: When counting tracks, all *reconstructed* tracks are considered, no special filter is applied. For the vertices, the *simulated* vertices are counted, again with no special cut applied.

3.1 Event topology of $c\bar{c}$ jets

8903 events have been analysed. Fig. 16 shows one such event. Association between the simulated and the reconstructed vertices has been performed on a by-tracks basis. This means that a reconstructed vertex is associated to the simulated vertex with which most tracks are in common. The simulated and reconstructed tracks are, in turn, associated "by hits", using the framework's default TrackAssociatorByHits. So, in order for a reconstructed track to be assigned properly, it has to share more hits with the correct simulated track than with any other track in the sample, independent of the absolute number of shared hits. The multiplicities of reconstructed tracks of primary and secondary vertices are shown in Fig. 4, the distances between primary vertices and secondary vertices in Fig. 5, and the number of reconstructible secondary vertices per event in Fig. 6. A reconstructible vertex is defined as having at least two associated reconstructed tracks. The Monte Carlo index has been used to distinguish between primary and secondary vertices. Only tracks within the reconstructed jet cones have been considered.

Note that the track multiplicities of the secondary vertices are the multiplicities of the sum of all secondary vertices in all reconstructed jets. The track multiplicity of a charmed meson is between two and three.

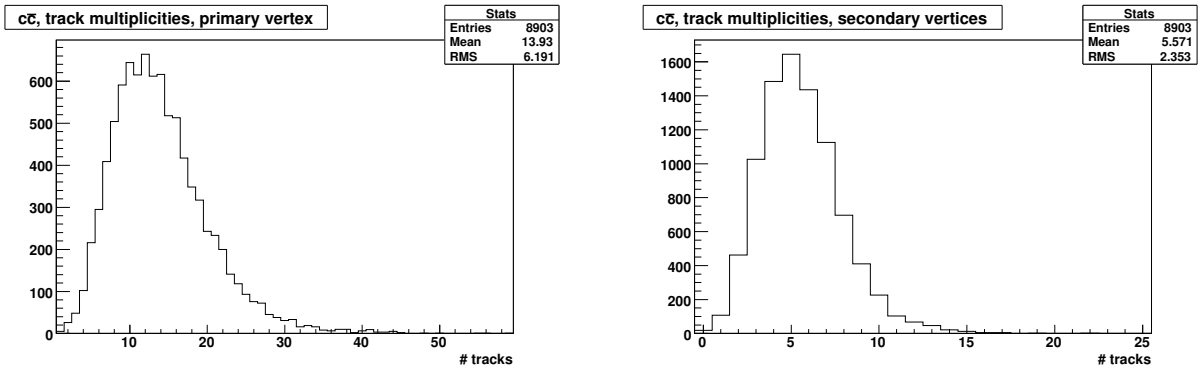


Figure 4: Reconstructed track multiplicities in the $c\bar{c}$ sample — primary vertex (left) and secondary vertices (right). Note that the track multiplicities of the secondary vertices are the multiplicities of the sum of all secondary vertices per event.

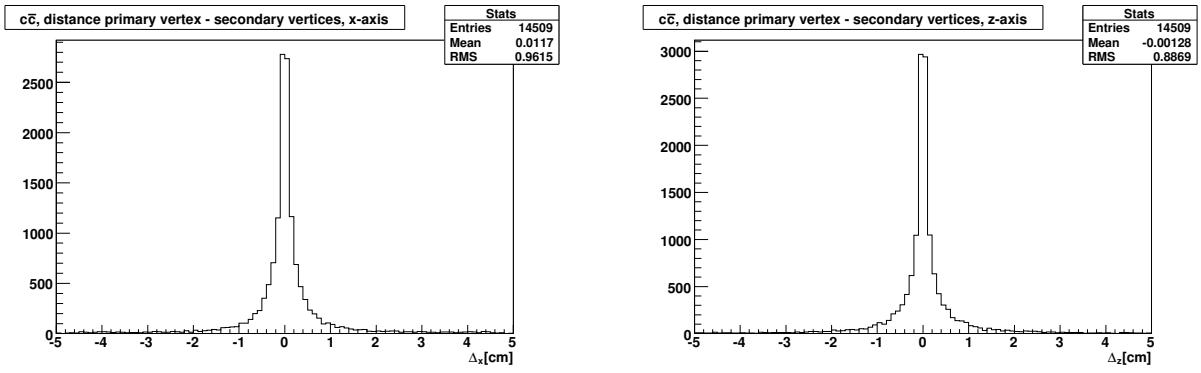


Figure 5: Distances between simulated collision (primary) vertices and decay (secondary) vertices in the $c\bar{c}$ sample.

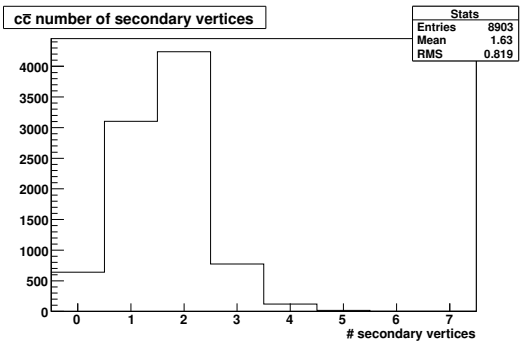


Figure 6: Number of reconstructible secondary vertices in the $c\bar{c}$ sample.

3.2 Event topology of $q\bar{q}$ jets

8936 events have been analysed. Only tracks within the reconstructed jet cones are considered. The track multiplicities of primary and secondary vertices are shown in Fig. 7, the distances between primary vertices and secondary vertices in Fig. 8, and the number of reconstructible secondary vertices per event in Fig. 9. Again, the track multiplicities of the secondary vertices are the multiplicities of the sum of all secondary vertices per event – summing over all reconstructed tracks in all jets.

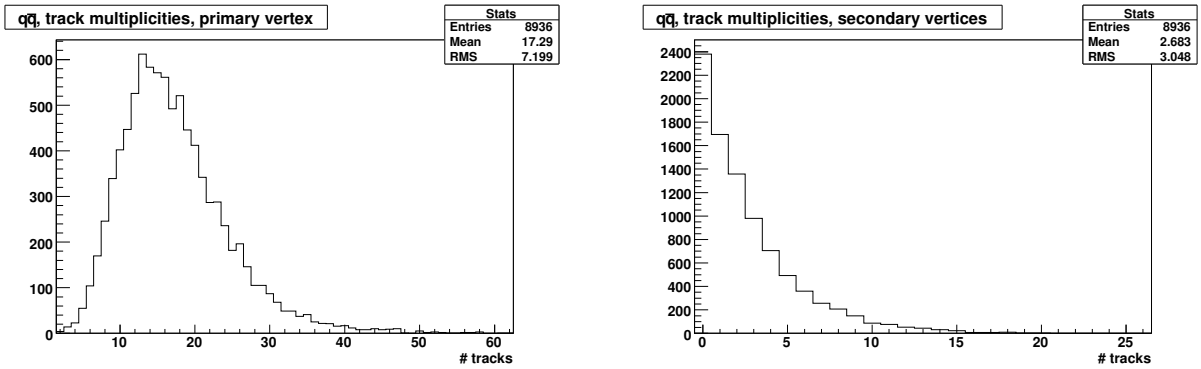


Figure 7: Reconstructed track multiplicities in the $q\bar{q}$ sample — primary vertex (left) and secondary vertices (right). Note that the track multiplicities of the secondary vertices are the multiplicities of the sum of all secondary vertices per event.

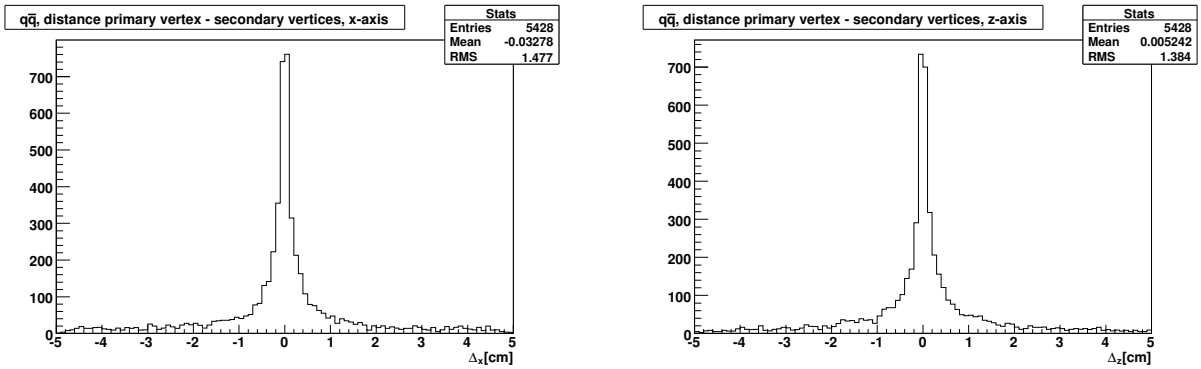


Figure 8: Distances between simulated collision (primary) vertices and decay (secondary) vertices in the $q\bar{q}$ sample.

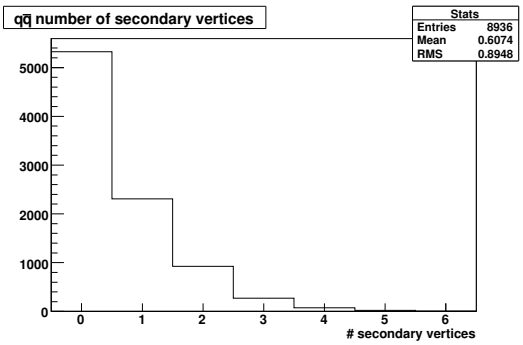


Figure 9: Number of reconstructible secondary vertices in the $q\bar{q}$ sample.

3.3 Kinematics of $\tau \longrightarrow \pi\pi\pi$

6404 events have been analysed, 5110 of which have all three π's reconstructed. Figure 10 shows the sums of the p_{T} of the three reconstructed decay tracks. τ leptons. The remaining 1294 events have only two reconstructed π's which were assignable to the corresponding simulated track. As the track association criterion, the framework's TrackAssociatorByHits with default values was employed – see Sec. 3.1 for a short description of the associator.

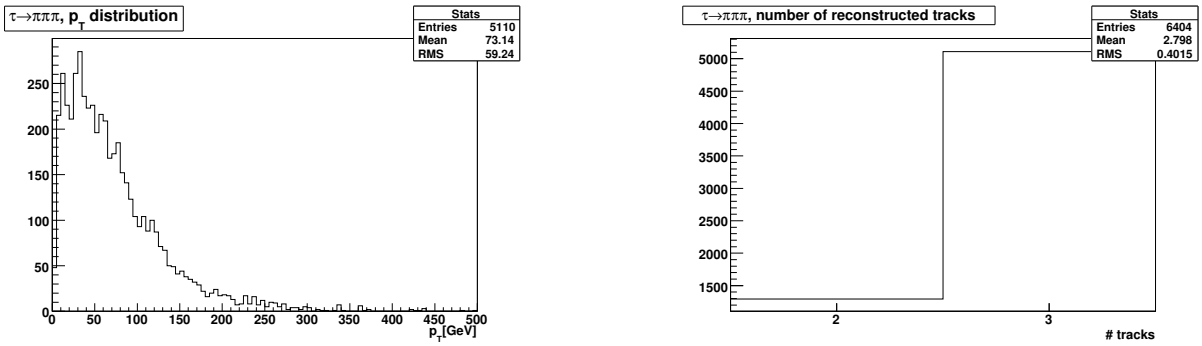


Figure 10: p_{T} distribution of the τ lepton (fully reconstructed decays only) and multiplicity distribution of the number of reconstructed tracks.

3.4 Kinematics of $B_s \longrightarrow J/\psi\ \varphi \longrightarrow K^+K^-\mu^+\mu^-$

9803 events have been analysed. 7451 events have all four tracks reconstructed "correctly": all four of them are assigned to the corresponding simulated track — for the details of the assignment criterion see, again, Sec. 3.1. In 2088 cases one track was not reconstructed correctly in the above sense; the analysis was performed with only three reconstructed secondary tracks. In 264 cases two tracks are missing. Figure 11 shows the p_{T} distribution of the $J/\psi\ \varphi$ system for the fully reconstructed events.

4 Technical aspects of the adaptive method

This section describes the various parts of the adaptive method. The technical choices that had to be taken will be presented and justified.

4.1 Track (re-)linearization

No matter what track parametrization is used, a charged track in a magnetic field can not be described exactly by a linear model. In order to deal with this non-linearity, the exact track model is approximated by a linear model. The linear expansion is recomputed if the estimated vertex has moved too far from the expansion point. For CPU performance reasons, track relinearization should only be performed when needed. The current implementation recomputes the linear approximation when the current vertex estimate moves by more than a certain threshold in

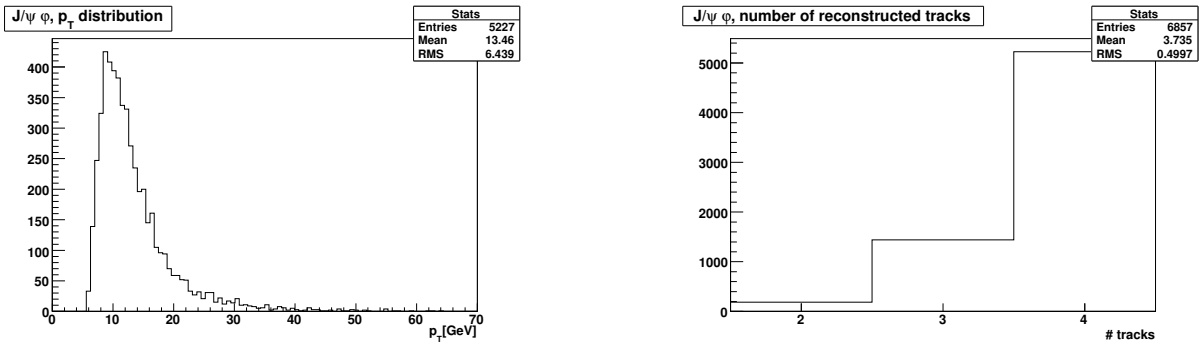


Figure 11: p_{T} distribution of the $J/\psi\ \varphi$ system (fully reconstructed decays only) and multiplicity distribution of the number of reconstructed tracks.

the transverse plane. The default for this threshold is currently at $100\,\mu\mathrm{m}$. Another possibility is hinted at in [10]: the definiteness of the matrix of second derivatives of the model can be used to determine whether the current estimate is still in the domain of attraction of the global maximum. Further studies in this direction are desirable.

4.2 Initial estimate of vertex position

A robust initial estimate of the vertex location is important in the adaptive estimation. It not only defines around which points the tracks are linearized, but also the initial assignment probabilities (weights) of the tracks. If the adaptive method is interpreted as an optimization procedure, then the initial estimate can be seen as its global aspect. It is imperative that it resides close to where the global optimum of the adaptive estimate is. The method by which it is produced must therefore be robust with a high break-down point [11].

4.2.1 The default algorithm

The input for all linearization point finders is a container of reconstructed tracks. The output is a point in three-dimensional Euclidean space. Details of the implementation are given in the appendix.

Many different algorithms have been tried. For the sake of brevity we shall in this note restrict ourselves to the presentation of the default method: the fraction-of sample mode with weights (FSMW, [12]), and show a comparison with a few other methods that have been tried [5]. This default method is based on the *crossing points* of the tracks. A crossing point is defined as the algebraic mean of the two points of closest approach of two tracks. To a crossing point we attach a weight which is a function of the inverse distance of the two tracks, such that a smaller distance between the two tracks gives a larger weight to their crossing point.

The weight function reads:

$$w = (d + d_{\min})^n \tag{10}$$

where d denotes the distance between the points of closest approach. The default values are: $n = -.5$, $d_{\min} = 10\,\mu\mathrm{m}$.

The FSMW finds the mode (point of highest density) of the crossing points, separately in each of the three spatial coordinates. Each one-dimensional mode finding starts by finding the smallest "weighted" interval that covers at least f percent of all data points, where f is a parameter of the algorithm. A weighted interval is defined as the length of the interval divided by the sum of all weights of the contained points. The procedure is iterative: it is recursively applied to the previously found interval, until an interval with two points remains. Finally, the mode of this particular spatial coordinate is the average of the remaining two points. Applying this iterative procedure separately to each spatial coordinate results in the final three-dimensional mode of the crossing points. The default value of f is 0.4 in the current implementation.

For performance reasons, the weights of the crossing points are ignored until the number of data points drops below a certain threshold. The default value for this threshold is 5. CPU performance was also the reason behind the decision to implement an upper limit for how many crossing points are considered. Since there is one crossing point for each track pair, their total number grows quadratically with the number of tracks. The default value of the upper limit is 400. If more track pairs are available, the most "interesting" ones are chosen, "interesting" being defined by

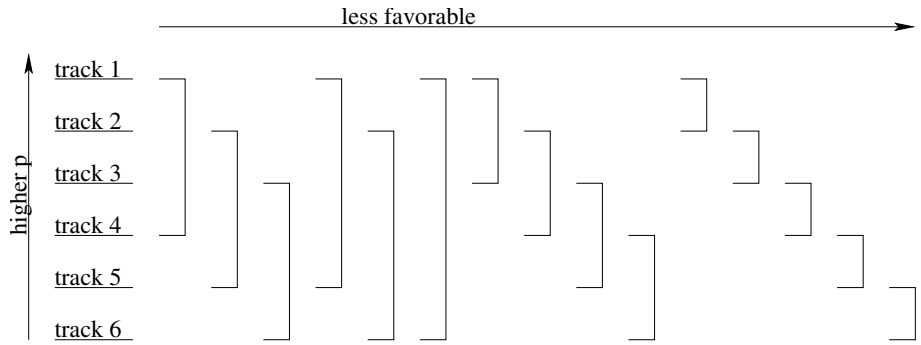


Figure 12: The order of track pairs considered in the crossing point based algorithms, shown for six tracks.

(a) using as many different tracks as possible, and

(b) as high-energetic tracks as possible (the length of the full 3d track momentum vector is currently used),

(c) mixing as much as possible high-energy tracks with low-energy tracks, as far as this is compatible with (a) and (b).

Fig. 12 shows the order of track pairs considered for the special case of six tracks.

4.2.2 Performance analysis of the FSMW

The performance of the FSMW method has been analysed and compared against a few other algorithms that are described in [5]. The results are summarized in Table 1. The column labelled with "RMS" denotes the RMS of the resolution plot of the z coordinate of the initial vertex estimate. The z coordinate is used because the differences between the various methods are particularly pronounced in this variable. "σ_{Fit}" refers to the standard deviation of a Gaussian distribution fitted into the resolution distribution. A least-squares fit has been used, assuming Gaussian errors on the uncertainty in each bin. "$> 2\,\text{mm}$" denotes the failure to find a linearization point whose z coordinate is within $2\,\text{mm}$ from the true vertex position. Note that the given "RMS" as well as "σ_{Fit}" refer to the distributions that have been truncated according to the "failure" criterion, i.e. at $2\,\text{mm}$. Finally, the column labelled "t" denotes the average time spent per event, in milliseconds, on a 2.8 GHz Intel Celeron processor. FSMW, the default algorithm, performs well in all scenarios. Its CPU consumption is also acceptable. Very notable is also the fact that the non-iterative Least Median of Squares (LMS) fails in high-multiplicity events. Fig. 13 shows two resolution plots of the default linearization point finder.

	$c\bar{c}$ (primary vertex)				$q\bar{q}$ (primary vertex)			
LinPtFinder	RMS $[\mu\text{m}]$	σ_{Fit} $[\mu\text{m}]$	$> 2\,\text{mm}$ ‰	t $[ms]$	RMS $[\mu\text{m}]$	σ_{Fit} $[\mu\text{m}]$	$> 2\,\text{mm}$ ‰	t $[ms]$
Other method (ISMS)	86	43	5	7.7	80	39	5	9.3
Other method (HSM)	90	47	2	3.7	85	41	3	5.7
Other method(LMS)	373	63	66	3.6	358	51	88	5.6
FSMW (default)	92	49	3	4.4	88	44	3	5.8

	$\tau \to \pi\pi\pi$ (secondary vertex)				$J/\psi\ \varphi \to K^+K^-\mu\mu$ (secondary vertex)			
LinPtFinder	RMS $[\mu\text{m}]$	σ_{Fit} $[\mu\text{m}]$	$> 2\,\text{mm}$ ‰	t $[ms]$	RMS $[\mu\text{m}]$	σ_{Fit} $[\mu\text{m}]$	$> 2\,\text{mm}$ ‰	t $[ms]$
Other method (ISMS)	630	458	146	0.1	290	80	41	0.5
Other method (HSM)	632	455	146	0.2	293	84	41	0.5
Other method(LMS)	632	455	146	0.2	554	272	154	0.3
FSMW (default)	617	436	140	0.2	289	85	40	0.5

Table 1: Resolutions and failure rates of different LinearizationPointFinders for the four test samples. See the text for detailed description.

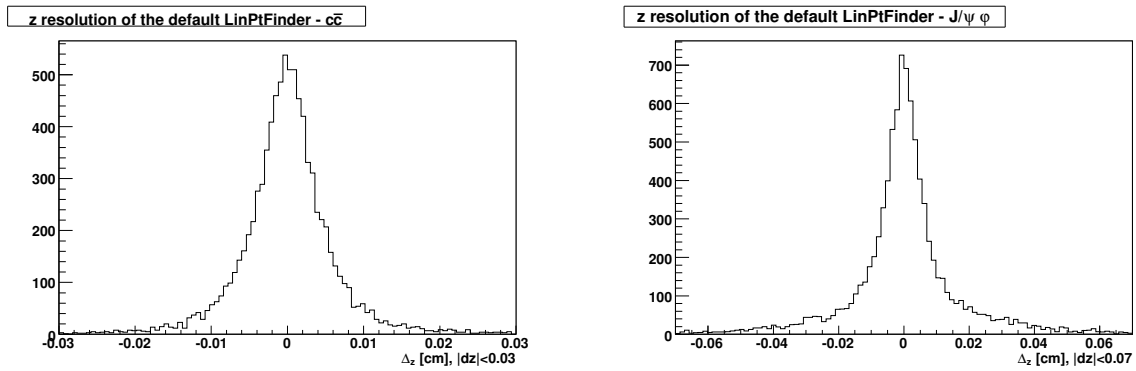


Figure 13: Resolution plots of the default LinearizationPointFinders.

4.2.3 Influence of the linearization point on the final estimate

It is interesting to study the importance of the initialization of the adaptive fitter. To this end the default linearization point finder was compared against three "artificial" finders:

- A Monte Carlo based finder that uses the simulated vertex as the fitter's initialization ("MonteCarlo"),
- a finder that always returns the point $(0, 0, 0)$ ("Zero"), and
- a finder that returns the result of the linear least-squares fitting method as the linearization point.

The results are given in Table 2. No beam spot constraints were applied. The columns match the ones given in Table 1. For the final fit the default adaptive vertex fitter was employed ($\chi_c = 3.0, T = 256, 64, 16, 4, 1, 1, \ldots$).

It can be seen that the initialization indeed *does* matter. The "zero" linearization point finder scores poorly. The comparison between FSMW and the linear method is interesting insofar as the linear method (which itself was initialized with the FSMW method) is mathematically equivalent to starting the adaptive fitter by assigning equal weights to all tracks. It can be seen that this leads to an increase of the residual tails. Note also that in the $c\bar{c}$ sample the adaptive fit with the simple "Zero" linearization point finder takes longer than the fit with the sophisticated FSMW. The reason is that a better linearization point speeds up the fitting procedure because fewer iterations are necessary for convergence. The comparison of FSMW with MonteCarlo indicates that there might be some space for improvement, albeit not very much.

	$c\bar{c}$ (primary vertex)				$q\bar{q}$ (primary vertex)			
LinPtFinder	RMS $[\mu\text{m}]$	σ_{Fit} $[\mu\text{m}]$	$> 2\,\text{mm}$ ‰	t $[ms]$	RMS $[\mu\text{m}]$	σ_{Fit} $[\mu\text{m}]$	$> 2\,\text{mm}$ ‰	t $[ms]$
Zero	191	28	110	18.9	194	26	115	25.1
MonteCarlo	72	30	1	12.1	53	28	2	15.7
KalmanVertexFitter	78	30	11	25.4	88	27	14	34.9
FSMW	72	30	2	16.7	55	27	2	21.4

	$\tau \to \pi\pi\pi$ (secondary vertex)				$J/\psi\ \varphi \to K^+K^-\mu\mu$ (secondary vertex)			
LinPtFinder	RMS $[\mu\text{m}]$	σ_{Fit} $[\mu\text{m}]$	$> 2\,\text{mm}$ ‰	t $[ms]$	RMS $[\mu\text{m}]$	σ_{Fit} $[\mu\text{m}]$	$> 2\,\text{mm}$ ‰	t $[ms]$
Zero	726	663	183	2.2	453	158	281	6.7
MonteCarlo	591	348	106	1.9	260	72	29	3.8
KalmanVertexFitter	590	354	114	4.7	262	72	31	8.0
FSMW	589	352	114	1.9	261	72	36	4.0

Table 2: Influence of the linearization point on the final (adaptive) vertex fit. See the text for further explanations.

4.3 Annealing schedule

A few annealing schedules have been tried out. Table 3 compares some of them in the four event topologies. As before, no beam spot constraints were applied. Again, the RMS and the (Gaussian) fitted σ of the z-coordinate are given. The label "$> 2\,\mathrm{mm}$" denotes the failure to reconstruct a vertex whose z coordinate is within $2\,\mathrm{mm}$ from the true vertex position, including truly failed fits (in which cases exceptions were thrown). The "t" column lists the average time spent per event, given in milliseconds. The "..." refers to geometric annealing schedules with an annealing ratio $r = 2$. The time was measured on a 2.8 GHz Intel Celeron processor and an annealing schema of $T = (256, 64, \dots)$ has been chosen as the default.

	$c\bar{c}$ (primary vertex)				$q\bar{q}$ (primary vertex)			
Schedule	RMS $[\mu\mathrm{m}]$	σ_{Fit} $[\mu\mathrm{m}]$	$> 2\,\mathrm{mm}$ ‰	t $[ms]$	RMS $[\mu\mathrm{m}]$	σ_{Fit} $[\mu\mathrm{m}]$	$> 2\,\mathrm{mm}$ ‰	t $[ms]$
1	84	30	1	13.2	68	28	2	17.3
4 3 2 1	79	30	2	14.1	59	27	2	18.2
8 4 ...	75	30	2	16.0	58	27	2	21.8
32 16 ...	73	30	2	16.7	59	27	2	21.4
256 64 16 4 1	72	30	2	17.3	55	27	2	22.2
512 256 ...	76	30	3	20.4	61	27	3	27.5
2048 1024 ...	74	30	2	22.7	65	27	2	29.8
8192 4096 ...	74	30	2	25.3	68	27	2	33.2

	$\tau \to \pi\pi\pi$ (secondary vertex)				$J/\psi\ \varphi \to K^+K^-\mu\mu$ (secondary vertex)			
Schedule	RMS $[\mu\mathrm{m}]$	σ_{Fit} $[\mu\mathrm{m}]$	$> 2\,\mathrm{mm}$ ‰	t $[ms]$	RMS $[\mu\mathrm{m}]$	σ_{Fit} $[\mu\mathrm{m}]$	$> 2\,\mathrm{mm}$ ‰	t $[ms]$
1	583	358	102	1.2	270	73	38	2.4
4 3 2 1	587	364	105	1.4	268	73	36	3.8
8 4 ...	586	347	106	1.8	269	72	36	3.8
32 16 ...	589	354	109	1.6	270	73	36	3.9
256 64 16 4 1	589	352	114	1.7	261	72	36	4.5
512 256 ...	589	356	116	2.5	261	72	35	5.1
2048 1024 ...	589	356	117	2.6	262	72	36	6.6
8192 4096 ...	590	357	117	2.9	262	72	36	7.2

Table 3: The choice of the annealing schedule influences the result. The "..." refer to geometric annealing schedules with $r = 2$.

4.4 Choosing a χ^2_{c}

Also a good default χ^2_{c} criterion needed to be found. To this end the same procedure as before has been applied: RMS, σ_{Fit}, fraction of outliers, and CPU time have been evaluated as a function of χ^2_{c}. The results, again, without any beam spot constraints, are shown in Table 4. The effect is more pronounced in the $c\bar{c}$ and $q\bar{q}$ sample, because of the larger number of outliers (secondary tracks). In the samples with small track multiplicity the results hardly depend on the choice of χ^2_{c}. Fig. 14 shows the same information, only in a more visual form. In the end a default value of $\chi_{\mathrm{c}} = 3$ has been chosen.

4.5 Prior information on the vertex position

A vertex fit can also make use of a prior knowledge of the vertex. This prior information is used as a linearization point with finite errors. The number of degrees of freedom of the reconstructed vertex is raised by three. The adaptive fitter can deal with such a prior information. One use case for this feature is to feed a fitter with the knowledge of the beam profile. This makes sense if it is known that the vertex that is to be fitted is a primary vertex. The analyses shown in this paper do not exploit any such prior information.

	$c\bar{c}$ (primary vertex)				$q\bar{q}$ (primary vertex)			
χ_c	RMS $[\mu m]$	σ_{Fit} $[\mu m]$	$> 2\,\text{mm}$ ‰	t $[ms]$	RMS $[\mu m]$	σ_{Fit} $[\mu m]$	$> 2\,\text{mm}$ ‰	t $[ms]$
1.0	77	30	2	18.3	57	28	2	24.5
2.0	74	30	2	17.7	56	28	2	21.8
2.5	74	30	2	17.7	56	28	2	21.8
3.0	72	30	2	16.3	55	27	2	22.4
3.5	72	30	2	16.3	55	27	2	22.4
4.0	76	31	2	16.4	55	28	3	21.7
5.0	76	32	3	16.0	61	28	3	20.9
6.0	83	33	2	15.4	64	29	3	21.4
7.0	86	34	3	16.0	66	29	3	21.1
8.0	88	36	3	15.8	68	30	2	20.4
9.0	91	37	3	15.8	72	30	2	20.4

	$\tau \to \pi\pi\pi$ (secondary vertex)				$J/\psi\ \varphi \to K^+K^-\mu\mu$ (secondary vertex)			
χ_c	RMS $[\mu m]$	σ_{Fit} $[\mu m]$	$> 2\,\text{mm}$ ‰	t $[ms]$	RMS $[\mu m]$	σ_{Fit} $[\mu m]$	$> 2\,\text{mm}$ ‰	t $[ms]$
1.0	595	363	116	2.1	269	75	39	5.3
2.0	593	354	115	2.0	266	73	37	4.6
2.5	593	354	115	2.0	266	73	37	4.6
3.0	589	352	114	1.9	261	72	36	4.2
3.5	589	352	114	1.9	261	72	36	4.2
4.0	591	345	116	2.2	256	72	34	4.1
5.0	595	351	117	1.6	257	73	35	4.3
6.0	600	365	118	1.7	261	75	35	4.4
7.0	599	359	120	1.8	265	75	34	4.4
8.0	601	361	120	1.8	264	75	34	4.7
9.0	602	359	121	1.6	267	76	34	4.5

Table 4: Results of the fit, as a function of χ_c.

4.6 Exceptions

The AdaptiveVertexFitter throws an exception ("Supplied fewer than two tracks"), if the user supplies one or no tracks. The class also throws an exception ("fewer than two significant tracks"), if, after the iterative fit, fewer than two significant tracks were found. Significant in this context means that the weight is above a certain threshold, which defaults to 0.01.

5 Case studies

This section is dedicated to two use cases that are intended to further illustrate the algorithmic procedure. Sec. 5.1 shows how the associated track weights change in each iteration step. Sec. 5.2 studies the algorithmic behavior at its low-multiplicity limits.

5.1 Evolution of track weights in a $c\bar{c}$ event

Fig. 15 shows how the track weights change in each iteration step in the adaptive method for one particular $c\bar{c}$ event (Fig. 16). The eleven tracks from the primary vertex are contaminated with three tracks from secondary vertices, plus two more tracks that could not be associated to any vertex. It is interesting to note that in this particular topology the fitter down-weights one of the primary tracks in the beginning. Only when the bias on the fit from the mis-associated tracks decreases is the fitter capable of "deciding for" keeping this track.

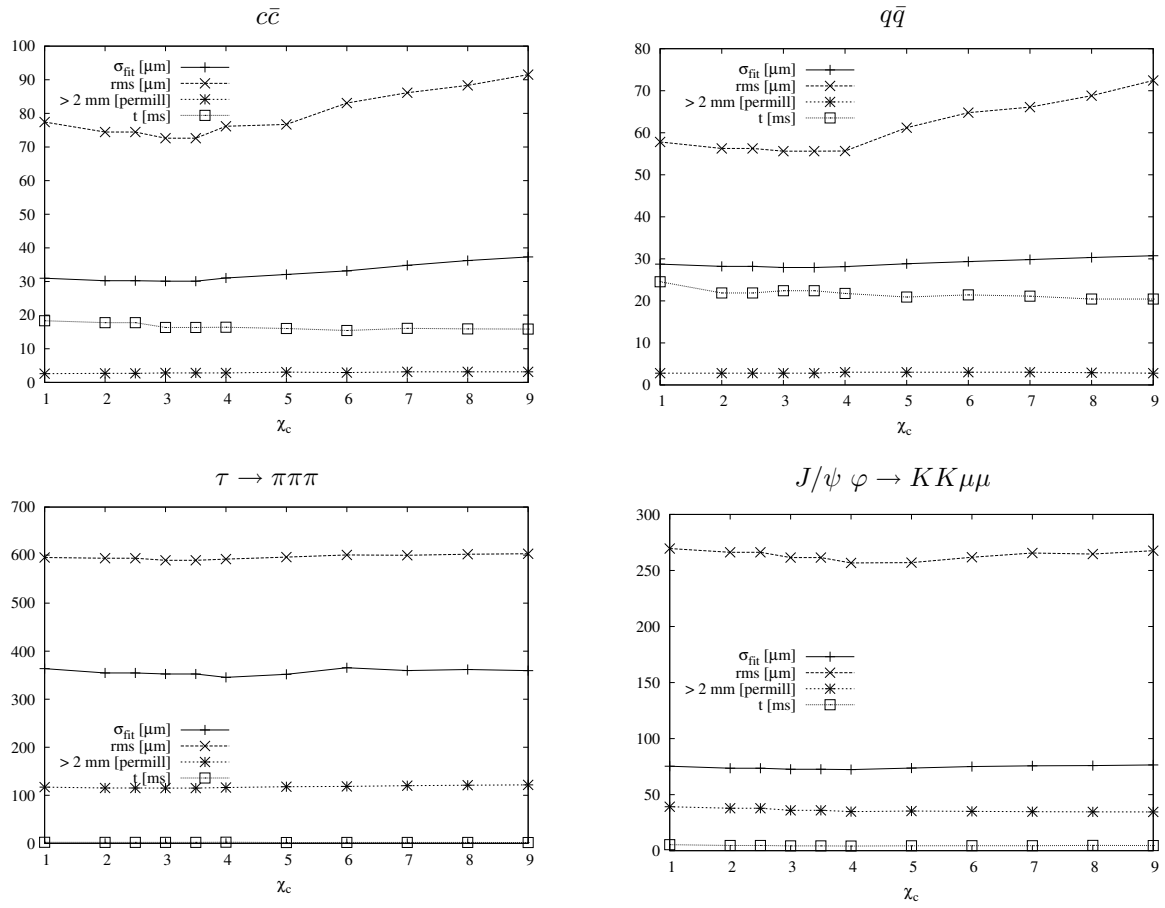


Figure 14: Fit results as a function of χ_c.

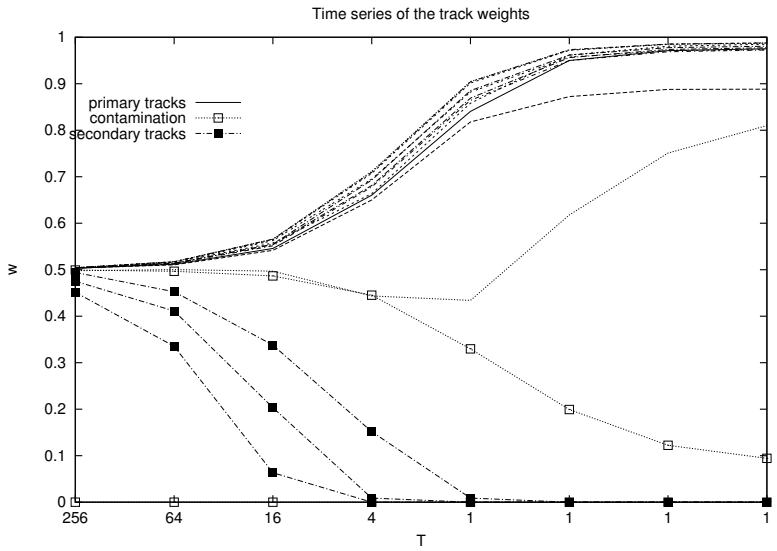


Figure 15: Evolution of the track weights of a $c\bar{c}$ primary vertex fit.

5.2 Track weights in a τ event

The adaptive method has originally been designed for high-multiplicity vertices with mis-associated tracks. It is thus interesting to study the behavior of the method in low-multiplicity vertices. To this end we investigate how the adaptive method behaves in cases of failure of the TrimmedKalmanVertexFitter (TKVF, see [6]), a least-squares fitter with iterative removal of incompatible tracks.

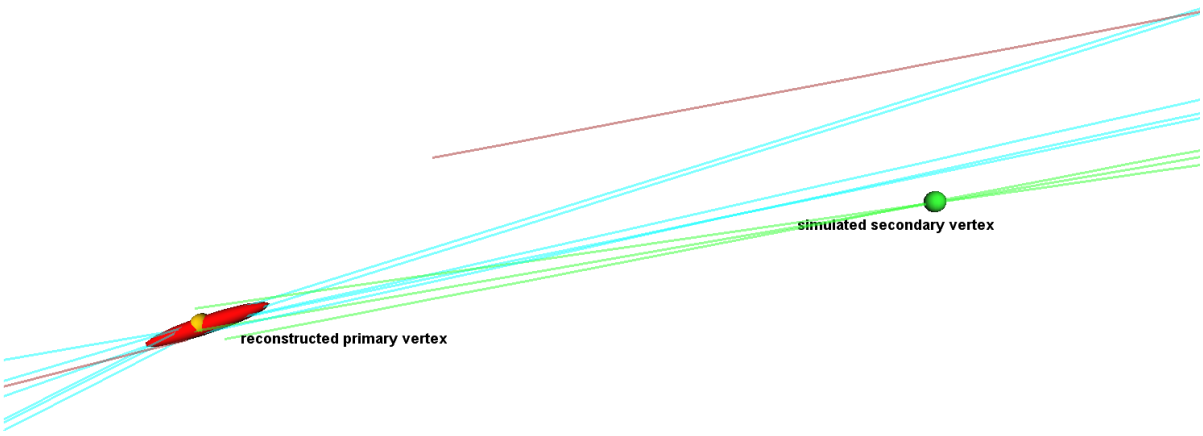


Figure 16: Snapshot of the $c\bar{c}$ event used in Sec. 5.1. The two "contamination" tracks and the three secondary vertex tracks are clearly visible. The ellipsoid represents the reconstructed vertex error. For visibility it is magnified by a factor of ten.

Fig. 17 shows the track multiplicities of the events in which the TKVF run with default values does not find a vertex. Fig. 18 shows the highest versus second highest track weights obtained by the default AVF, run on this τ-subsample (left hand plot). It can be seen that in the majority of the cases, the vertex is pulled towards a single track. Already the second highest track weight is zero or close to zero in most cases. The right hand plot of Fig. 18 shows how the second highest track weight $w_{(2)}$ affects the distribution of the standardized residuals of the fitted vertices' z coordinate. For the events with $w_{(2)} \approx 0$ the vertex errors tend to be over-estimated and the standardized residuals cluster near 0.0. This fact can also be seen in Fig. 19. The pronounced peak in the left plot comes from these "one-track" events. Introducing a cut on the second highest track weight of $w_{(2)} >= 0.01$ removes the peak (right plot), at the price of throwing some events away. Finally, Fig. 20 repeats the plots of Fig. 19 (fitted with the superposition of two Gaussians), only this time the complete event sample is used. This study justifies the choice of the minimum weight for a track to contribute significantly to the vertex (see Sec. 4.6).

One main advantage of the AVF over "hard-assigning" algorithms like the TKVF is particularly visible in this example. When given three mutually incompatible tracks, the TKVF cannot but fail. Not so the AVF. Since tracks are never fully discarded in this "soft-assigning" algorithm, a vertex can still be found. An *a-posteriori* decision of what to do with the vertex can be (and is) made, based on the track weights with respect to the final vertex.

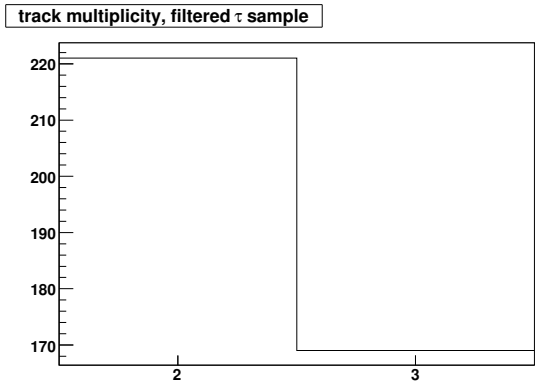


Figure 17: Track multiplicities in the τ sub-sample for which the TrimmedKalmanVertexFitter fails.

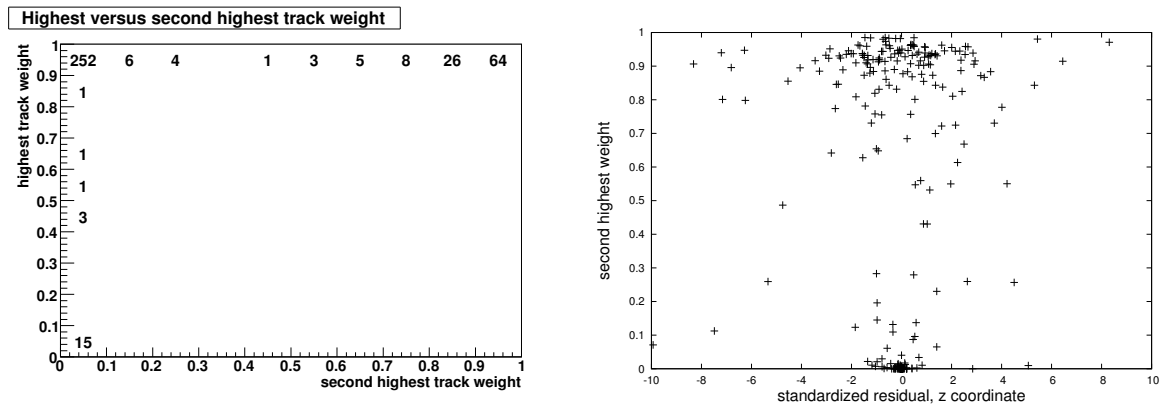


Figure 18: Track weights and standardized residuals of the AVF, in the τ sub-sample for which the TrimmedKalmanVertexFitter fails. The left plot shows the highest track weights plotted against the second highest track weights. On the right the second highest track weight is plotted against the residuals of the z coordinates of the vertices.

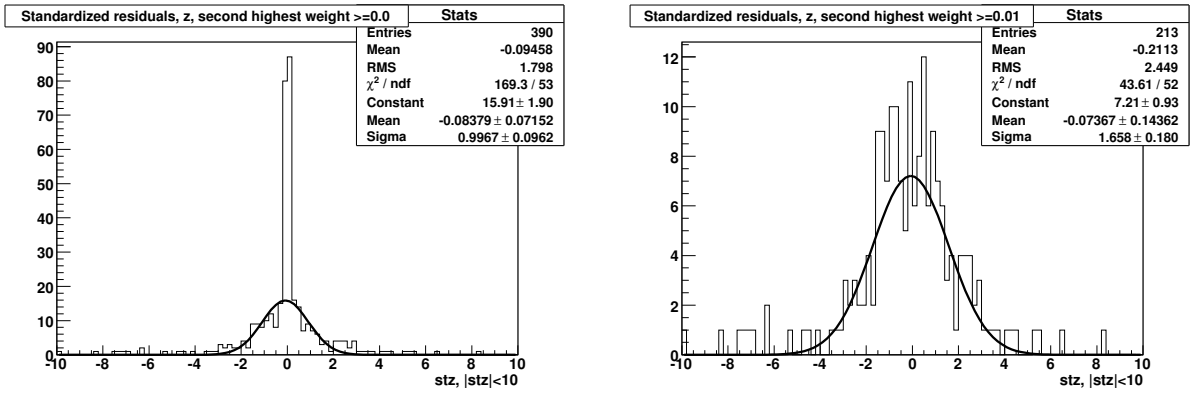


Figure 19: Standardized residuals, with all events of the τ sub-sample for which the TrimmedKalmanVertexFitter fails (left), introducing a threshold on the second highest track weight (right).

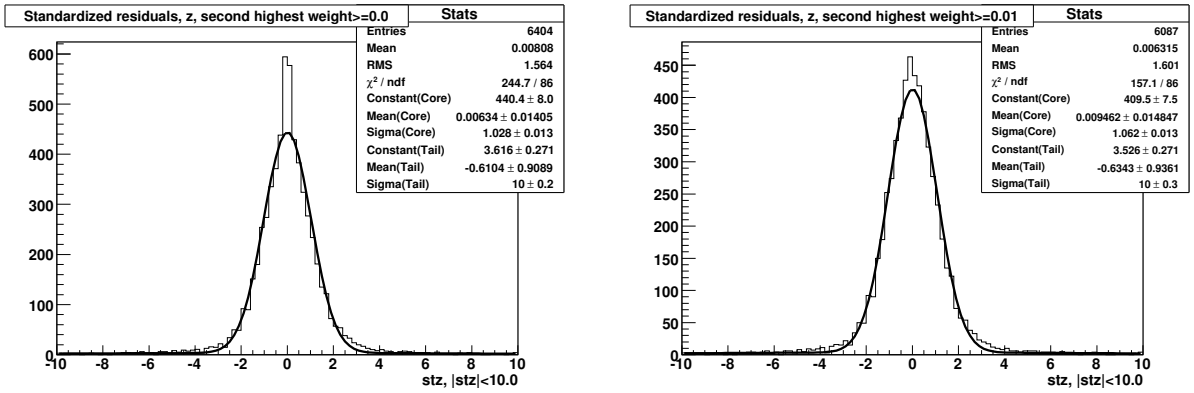


Figure 20: Standardized residuals of the AVF, for the full event sample, with (right) and without (left) a threshold on the second highest track weight.

6 Interpretation of the track weights

After fitting a vertex with a linear fitter, physicists usually discard vertices which fail a certain χ^2-probability cut. Only then one usually continues with the analysis. When using an adaptive fitter, the issue is more subtle. The χ^2 probability is not trivial to interpret; the information is now in the track weights, albeit at a more fine-grained level: A track with $w < 0.5$ is by construction an outlier; one with $w > 0.5$ is an inlier. "Cutting" at anything other than 0.5 is discouraged; it is statistically meaningless.

So the user now implicitly defines a cut on the tracks when choosing χ^2_c. It is equivalent to cutting at a certain track's χ^2 probability, knowing that an individual track contributes two degrees of freedom to the vertex fit.

So what should one really do with the final vertex, knowing that the "goodness of fit" information is hidden in the track weights? The authors believe that this a question of the specific use case. Consider the case of fitting $J/\psi\,\varphi \rightarrow KK\mu\mu$. Assume that the result of a fit is that three track weights are close to one, while the fourth weight is close to zero. The question of discarding the vertex is a question of what is relevant. If it is important that the vertex with its four daughter particles be reconstructed fully and correctly, then discarding this event is a possibility. If only the lifetime information of the mother particle is the relevant information, then the reconstructed vertex seems a perfectly legitimate candidate.

7 Summary and Outlook

Let us not throw away data all too hastily. Instead, let us weight and re-weight the data, consider and reconsider alternative models. Only if we must, at the latest possible stage, shall we distinguish between "in" and "out", discriminate between signal and noise.

The authors (a formal answer to Mr. Edgeworth, see p. 2)

The adaptive vertex fitter is a general-purpose algorithm that can be used in a very wide range of applications. Its most particular asset is the fact that no specific information on the type or level of contamination is required (see also [13]). This feature must be valued highly, considering the challenging LHC environment that has to be faced. Vertex fitting is used in a few high level tasks such as b-tagging or kinematic fitting. It is not yet clear which consequences the introduction of a soft track-to-vertex association will have on this higher level code. It can be expected, though, that the extra information that is contained in the track weights can be exploited also in these parts of the analysis.

Acknowledgements

The authors would like to thank the referees Danek Kotlinski and Ian Tomalin for their valuable comments and suggestions which have led to substantial improvements.

References

[1] R. Frühwirth. Application of Kalman filtering to track and vertex fitting. *Nuclear Instruments and Methods in Physics Research A*, 262:444, 1987.

[2] R. Frühwirth, P. Kubinec, W. Mitaroff, and M. Regler. Vertex reconstruction and track bundling at the LEP collider using robust algorithms. *Computer Physics Communications*, 96:189–208, 1996.

[3] CMS Collaboration. ORCA, CMS OO Reconstruction. `http://cmsdoc.cern.ch/orca`.

[4] CMS Collaboration. CMSSW, CMS SoftWare. `http://cmsdoc.cern.ch/cms/cpt/Software/html/General/`.

[5] W. Waltenberger. *Development of Vertex Finding and Vertex Fitting Algorithms for CMS*. PhD thesis, TU Wien, 2004. CMS TS-2006/12. See also `http://publications.teilchen.at/ww_diss.pdf`.

[6] T. Speer, K. Prokofiev, R. Frühwirth, W. Waltenberger, and P. Vanlaer. Vertex Fitting in the CMS Tracker. (CMS-NOTE-2006-032), 2006.

[7] M. Ohlsson, C. Peterson, and A. Yuille. Track finding with deformable templates — the elastic arms approach. *Computer Physics Communications*, 71:77, 1992.

[8] R. Frühwirth and A. Strandlie. Track fitting with ambiguities and noise: a study of elastic tracking and nonlinear filters. *Computer Physics Communications*, 120:197–214, 1999.

[9] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons, New York, 1986.

[10] H. Eichinger and M. Regler. Review of Track-Fitting Methods in Counter Experiments. Technical Report CERN 81-06, CERN, Geneva, 1981.

[11] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, New York, 1987.

[12] David R. Bickel and Rudolf Frühwirth. On a fast, robust estimator of the mode: Comparisons to other robust estimators with applications. *Computational Statistics & Data Analysis*, 50(12):3500–3530, August 2006. available at `http://ideas.repec.org/a/eee/csdana/v50y2006i12p3500-3530.html`.

[13] J. D'Hondt, R. Frühwirth, P. Vanlaer, and W. Waltenberger. Sensitivity of robust vertex fitting algorithms. *IEEE Transactions on Nuclear Science*, 51(5):2037–2044, 2004.

Appendix: Implementation details

Our implementation not only knows of data objects, but also of algorithm objects. A *VertexFitter* is an object that maps a set of reconstructed tracks on a reconstructed vertex, see Fig. 21. Furthermore, the sequential (weighted) update of a vertex candidate with a single track is encapsulated in its own class, the *VertexUpdator*. The algorithms that compute the first rough guess of the vertex location also have their abstract base class, called *LinearizationPointFinder*. Finally, also the recomputation of the track momenta after the vertex fit (the "smoothing" procedure) and the computation of the annealing temperature are encapsulated in separate classes.

The task of linearization point finding can be formulated on top of the crossing points, although other formulations are conceivable (see [5]). If crossing points are used, a *three-dimensional mode finder* such as the FSMW is employed to compute the location of the vertex candidate. The software design reflects this simple relationship between linearization point finders and mode finders, see Fig. 22.

The code originally developed for ORCA was ported to CMSSW. The UMLs (Figs. 21 and 22) are valid for the CMSSW implementation also, except for one tiny difference: RecTracks are now known as reco::Tracks.

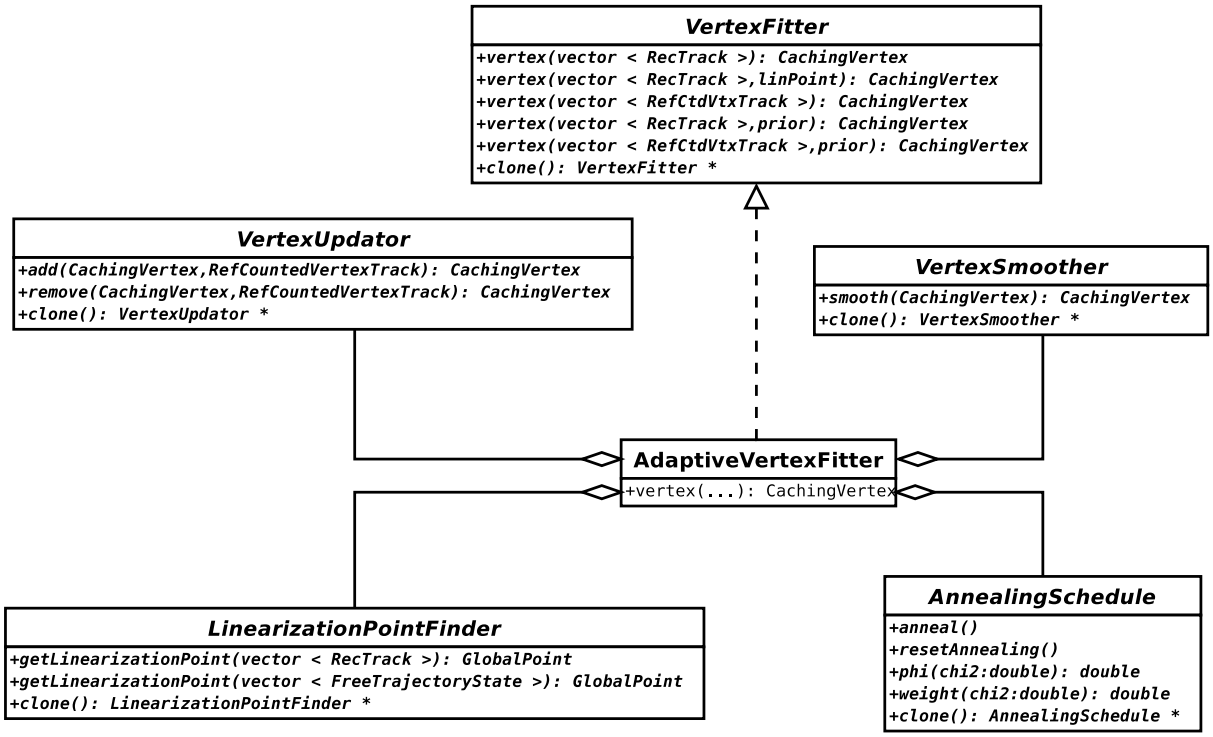


Figure 21: Implementation of the adaptive vertex fitter.

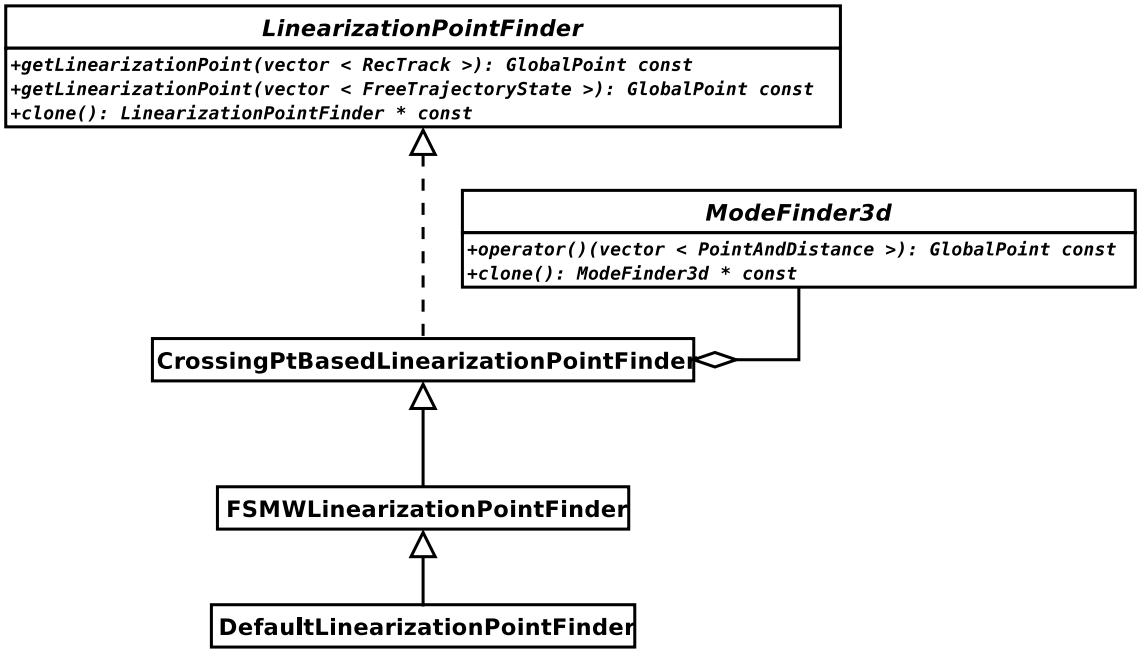


Figure 22: The DefaultLinearizationPointFinder and its inheritance.